In the Specification:

Please make the following changes to the patent specification for the paragraphs at:

Column 1, lines 10-15:

B1

U.S. Pat. No. 5,257,367 to Goodlander et al. for a Data Storage System With Asynchronous Host Operating System Communication Link, filed on Jul. 17, 1990 as U.S. patent application Ser. No. 544,127 and issued on Oct. 26, 1993 and assigned to the assignee of the present patent [application; and to] application.

Column 1, Line 64 - Column 2, Line 13

B2

Another technique is "striping", which is customarily employed as a technique for increasing the speed with which data may be written to or read from the disk drives of a mass storage system by distributing a body of data across a plurality of disk drives so that reads and writes from and to the disk drives with respect to segments of the data can be overlapped. Striping also serves to protect the data, in which role it is usually used together with parity, by storing [of] a body of data across a plurality of disk drives so that any data that is destroyed or damaged on one of the disk drives may be reconstructed from the remaining data and the parity information. An extreme embodiment of this technique is found, for example, in the Thinking [machines] Machines Corporation CM-2 system which operated with 39 bit words, each containing 32 data bits and 7 parity bits, and stored the bits of each word in parallel across 39 disk drives, with one bit of each word being stored on each drive.

Column 2, lines 37-43

B3

Mirroring the entire body of data and parity information stored in a mass storage system, in turn, immediately doubles the number of disk drives required to store the data and parity information and, again, the cost of the mass storage unit soon becomes

B3 Cont'd

impractical and the performance of the mass storage system, that is, the rate at which data may be written and read, may suffer.

Column 2, line 64 - Column 3, line 5

B4

According to the present invention, the mass storage system includes a plurality of mass storage devices for storing data and parity information wherein the system includes a host processor including memory and disk management facilities and a disk platform connected from the host processor and controlling a plurality of disk drive units comprising the mass storage devices, and a protection mechanism providing user selectable levels of protection against data loss.

Column 3, lines 48-54

B5

Another and independent aspect of the present invention provides user selectable levels of protection by allowing the writing of data blocks to the disk drives without corresponding parity information being stored in the disk drives. In this aspect of the invention, the memory management mechanism is responsive to assertion of a parity inhibit command for writing only data blocks into the disk drives.

Column 4, lines 9-31

B6

The data protection mechanisms of the present invention thereby allow a user to selectively tailor the level of protection in a mass storage system to meet the changing requirements of operation and use of the system by providing selectable levels of protection. For example, at the lowest level the system may be operated without mirroring and with parity generation and storing inhibited, thereby providing the maximum rate of data reception and storage but the lowest level of protection. At the next level, parity information may be generated and stored while the data is being received and stored, thus providing a higher level of protection but a lower data transfer rate. At the next level, the user may select intra-mirroring, wherein data that is selected and identified by the user to be additionally protected is written to logical units of the disk drives that have been selected and designated as mirrored logical units, thereby providing a still higher level of protection for the most critical or important data at the cost of some

increase in storage space. Finally, [the] using the same mechanism as provided for intra-mirroring but designating all logical units for mirroring, the user may achieve full mirroring for all data stored in the system, but at the cost of significantly increased storage space.

Column 4, lines 39-40

FIGS. 1(*a*) to [(*i*)] (*h*) show the historical development of the performance of computer systems;

Column 4, lines 41-42

[FIGS. 1(*a-g*), present] FIG 1(i) presents a simplified block diagram of a prior art approach to a disk-based memory system;

Column 4, lines 43-44

[FIGS. 1(*h*) and 1(*i*) illustrate an improved system in which the present invention could be implemented;]

Column 4, lines 53-54

FIG. 5 is a simplified block diagram of [the] one of the disk controller channels of the disk memory platform of FIG. 3;

Column 5, lines 4-6

A. General Description of Host Computer Systems With Associated Mass Storage Systems (FIGS. 1(*a*) through [1(*h*)] 1(*i*) and 2 through 6)

Column 5, lines 7-53

A given system's overall performance capability is the result of the integration of a number of independent technologies whose individual growth in performance/capability over time differs markedly from each other. FIGS. 1(*a*) to [(*g*)] (*h*) shows the historical performance of computer systems' underlying technologies. FIG. 1a shows the exponential growth in semiconductor component performance. The factors

behind this are well known and include advances in process technology that increase circuit density and speed. Shrinking geometry's and increased wafer yields combined with circuit design innovations mean semiconductor performance should continue its exponential growth. FIG. 1b reflects the exponential growth in CPU hardware performance as measured in MIPS. CPU's are the direct beneficiaries of semiconductors as well as circuit design improvements and architectural innovations such as massively parallel CPU's. FIG. 1c shows the trend in performance for operating systems. OS performance is being flattened by several factors such as the additions of user interfaces, graphics support, and the sheer growth in size over the years, which has made OS's one of the most voracious consumers of computer resources. FIG. 1d shows the capacity/performance improvements of disk drives. This curve could be best described as "leap-linear." Disk drive device performance and capacities tend to grow linearly until a new technological event occurs. Such events in the past have been the introduction of sealed disk Winchester technology in the early 70s, the introduction of thin film heads and plated media in the 80s, and, in the 90s, the general introduction of 5400 and 7200 RPM drives to cut latency delays and contact recording technology that may push track densities to 100,000 per inch. FIG. 1e demonstrates the performance of I/O systems. This curve represents the composite effect of CPU and controller hardware, operating systems, and disk drives. It should be noted that the exponential growth in semiconductor performance has not been reflected in I/O system performance, which has seen near linear growth. FIG. 1f reflects a commonly understood phenomena with applications programs that, over time, additions, changes, and maintenance to the program tend to lead to a decrease in its performance. FIG. 1g demonstrates that, [over all] overall systems performance has showed continued improvement, but at a much slower rate than its underlying technologies. In fact, without hardware upgrades and improvements [over all] overall system performance declines in response to the performance of operating systems and application programs.

Column 5, line 54 – Column 6, line18

FIG. [1(*h*)] 1(*i*) shows a typical prior art computer system employing disk drives for storage. The host computer 10 (i.e. the one interfacing with the computer operators)

includes an operating system 12. As known to those skilled in the art, the operating system is a set of computer programs that run continuously while the computer has its power on. The operating system controls all the functions of the computer including requests for operating portions of the memory, error response, and input/output (I/O) requests. The computer 10 has a disk controller 14 connected thereto and the disk controller 14, in turn, is connected to four disk drives 16. In use, an applications program (not shown) makes a request for data from the operating system 12. The location of the data is completely transparent to the applications program; that is, the applications program has no idea where the data is physically located. At system setup time (or possibly subsequently through operator input), the locations of the data [is] are stored in tables (not shown) which are part of or accessible by the operating system 12. Knowing from the tables that the requested data is on a particular disk drive 16 at a particular track between starting and ending sectors, the operating system 12 outputs a disk read request on line 18 to the disk controller 14. The disk controller 14, in turn, then issues a read request to the appropriate disk drive 16 on its connecting line 20 which causes the read head (not shown) within the disk drive 16 to move to the designated track and then read data and output it to the disk controller 14 on the line 20 from the starting sector to the ending sector. When the data has been received by the disk controller 14 (into an appropriate cache/buffer [memory,] memory), the operating system 12 is informed by an appropriate signal on line 18.

B13 Cntd

Column 7, lines 20-47

An improved overall computer system employing both disk and near line archival storage and typical of the systems in which the present invention may be employed is shown in FIG. 2 where it is generally indicated as 22. The system 22 has a host computer 10 containing an operating system 12 with its tables [24.] 25. There is also a console privilege interface 26 by means of which outside user consoles (not shown) can be used to access the host computer operating system 12. There is once again a disk controller 24 since there is no change to the operating system 12 and the operating system 12 is set up to interface with the disk controller 24. Rather than being connected directly to the disk drives 16 as in the prior art approach of FIG. 1(i), however, the single line 20 from the

B14

disk controller 24 is connected to an intelligent disk platform 28. The disk platform 28 is then connected to interface with the disk drives 16 through lines 30. Additionally, the disk platform 28 has a bi-directional connection 32 through a communications link 34 to the console privilege interface 26. In the preferred embodiment as applicable for large-scale storage systems, there is also near line archival storage apparatus 36 connected to the disk platform 28 through line 38. To perform within the system 22 of this invention, the near line storage apparatus 36 should be of an automated variety selected from any of a number well known in the art where off-line storage media are loaded for reading and writing on request by automated mechanisms so as to avoid the necessity of operator intervention.

Column 7, line 48 – Column 8, line 24

To accomplish its unique improvements over the prior art, the disk platform 28 includes its own computing capability as represented by the computer block 40. As will be seen shortly, the computer 40 may, in fact, comprise multiple processing units; but, for the present it is sufficient to note that the disk platform 28 is not the "dumb" controller 14 of the prior art. Employing the bi-directional connection 32 through the communications link 34 to the console privilege interface 26, the computer 40 can find the location of data from the tables [24] 25 of the operating system 12. The location of data within the disk drives 16 or the near line archival storage apparatus 36 is, therefore, transparent to both the operating system 12 and the applications programs requesting it. If requested data is located on the near line archival storage apparatus 36, it can be retrieved automatically and then be relayed to the operating system 12 just as if it was on one of the disk drives 16. More importantly, the preferred computer 40 is of the self learning variety which learns through experience. Thus, for example, if a particular file from the near line archival storage apparatus 36 is used at a given time and/or day each month (as in preparing a payroll), the logic of the computer 40 will learn that from experience and begin loading the file from the near line storage apparatus 36 in anticipation of the expected request so that when the request is made, the file is already read in and available for use. Thus, the overall system performance of the system 22 is not only improved over the prior art for a given level of file location transparency to the users; but, additionally,

the overhead drops as the system learns the repeated patterns of use associated with its users. In addition, whereas the prior art approach of FIG. 1(i) could only do system and equipment diagnostics by taking the computer 10 off-line or by increasing the complexity and overhead of the operating system 12 once again by having the operating system 12 perform such functions in a background mode, the storage system and equipment diagnostics are now performed by the computer 40 located within the disk platform 28 on a continuing and time-available basis. As will be seen from the description which follows, when the disk drives 16 have a fault or error, any errors and problems found can be corrected or at least [pin pointed] pinpointed for operator correction and replacement without taking the system 22 off line or causing any significant degrading of the performance thereof.

B15 Contd

Column 8, lines 39-60

In FIG. 3, line [18] 32 is labeled as the "SERIAL INTERFACE" and line 20 is labeled as the "HOST INTERFACE". In a tested embodiment, the SERIAL INTERFACE of line [18] 32 is an RS-232 interface and the HOST INTERFACE of line 20 is a SCSI (small computer system interface) interface. This choice was as a result of the availability of commonly-used equipment for use in testing only and those skilled in the art will readily recognize and appreciate that the same techniques being described here by way of example could be accomplished employing other hardware interface methods and apparatus known in the art, or yet to be developed. In this regard, the improvements of this invention are both universal and adaptable. The two lines are connected to [a] an interface and driver unit 42 which provides the host interface, serial interface, and LCD display driver functions as indicated therein. The logic and apparatus of interface and driver unit 42 is also connected to a display 44. The display 44 is not an absolute requirement; but, is preferred so that messages can be provided to an operator as, for example, in the event of a detected malfunctioning disk drive 16 which has been removed from operation and should be replaced in order to restore [fill] full system capability.

B16

Column 9, line 63 – Column 10, line 13

The CPU 60 includes an embedded array disk operating system 61 and employs its private memory 62 to keep track of the contents of the cache memory 56 so that it can respond optimally to requests from the host computer 10. The CPU 60 in this system only issues high level commands to the disk controller channels 46 so that the multiple, low-level command approach, which occupied valuable time on the CPU bus [66,] 52, is eliminated. The micro-processors 50 each contain firmware that not only performs the actual low-level command steps required to do disk transfers; but, in addition, performs continuing self-testing of the individual controller channel on a time-available basis to assure that the components are continuing to work properly. Should such self-check indicate a potential problem, the CPU 60 then conducts further testing of the subject disk controller channel 46 to see if an on-line "hot" spare disk drive 16 or disk controller channel 46 should be employed while the malfunctioning unit is flagged for work on the display 44 and removed from use temporarily.

B17

Column 10, lines 21-47

To get a better idea of the operation of the disk platform 28 of this invention, we will now turn to FIGS. 4 and 5 with particularity. FIG. 4 shows further details of the interface and driver unit generally labeled as 42 while FIG. 5 shows further details of one of the disk controller channels 46. With reference first to FIG. 4, there is an XBUS driver 66 connected to the XBUS 64 with a connector 68. There is also a host interface driver 70 (SCSI in the tested embodiment) connected into line 20 back to the host computer 10 by a connector 72. As with the other elements, there is also a local micro-processor 74 to control the operation of the elements of the interface and driver unit 42. The micro-processor 74 interfaces with a display driver 76 (which is connected to the display 44) and a serial interface driver 78 (which is connected to the serial interface on line [18).] 32). All the driver elements are well known to those skilled in the art and will be chosen according to the type of device with which they must interface. The micro-processor 74 is also connected to the CPU bus 52 with connector 80. The heart of the interface and driver unit 42 and most important part thereof is a pair of unidirectional FIFOs 82 and 84. Under the control of the local micro-processor 74, FIFO 82 receives and transmits data

B18

from the XBUS 66 to the host computer 10. Similarly, FIFO 84 receives and transmits requests and data from the host computer 10 to the XBUS 66. Thus, bi-directional transmissions can take place between the XBUS 66 and the host computer 10. This is another feature of this embodiment which improves the overall throughput of the system 22.

Column 10, lines 48-67

The disk controller [channels] channel 46 depicted in FIG. 5 also includes an XBUS driver 66 and a disk drive interface driver 92 connected to the associated disk drive 16 with their associated connectors 68, 94. Similarly, the local micro-processor 50 is connected to the CPU bus 52 with a connector 80. In addition to the data buffer memory 48, there is a buffer address register 86 which controls the locations in the memory 48 which are accessed and a data traffic semaphore 88 which operates in a manner readily apparent to those skilled in the art to control access to and from the memory 48 under the control of the micro-processor [58.] 50. Thus, it is the data traffic semaphore 88 which actually inserts the addresses into the buffer address register 86. The data traffic semaphore 88 must be present to prevent simultaneous access to the memory 48 by both the XBUS 64 (and elements connected therethrough) and the host computer 10. Such devices are well known and employed in the computing art for that purpose as any attempt to simultaneously read from and write into a memory location can cause irreversible errors.

Column 11, lines 1-11

The near line archival storage channel [100] 130 is controlled in the same manner as disk controller channel 46 through microprocessor 50 and cache/buffer memory 48 and contains the logic to control by way of control bus 101 the near line archival storage 103 and its individual elements 104, 105 and [106] 109 to read and write data by way of data bus 102. Data read from near line archival storage 103 is held in cache memory 56 or on disk drives 16 and is accessed by the host computer with sector numbers beyond the physical limits of disk drives 16 creating a virtually boundless storage capacity.

Having thus described the construction and operation of the system 22 in general, a more specific example of its unique mode of operation will now be described with reference to FIG. 6. For simplicity, FIG. 6 depicts in simplified form only the cache/buffer memories 48 in the channels and the cache memory 56 as connected by the XBUS 64. Assume that a request has been made by the host computer 10 to read data. The disk platform 28, of course, knows (or can determine) the location of the data in the disk drives 16 through its above-described access to the tables [24] 25 in the host computer 10. According to fault tolerant techniques, the data (and its parity bits) are spread across the disk drives 16. From the contents of its private memory 62, the logic in the CPU 60 knows the present contents of the cache memory 56. Anything that is already in the cache memory 56 will not be re-read, of course, which is the usual function of any cache memory (i.e. to eliminate redundant and unnecessary disk accesses). The CPU 60 then issues high level requests to the various disk controller channels 46 to have them retrieve the elements of the data from their locations on the disk drives. The requests also go to the cache memory & control unit 54 so that it knows what is going on. From there on, the collection of the data and its transfer to the host computer 10 is under the control of the micro-processor 58 in the cache memory & control unit 54. The micro-processor 58 assigns available buffer space (as indicated by the dashed box 90) in which to receive the data of the request. The data segments are asynchronously brought into the buffer memories 48 under the control of the micro-processors 50 as a function of the originally-issued high level commands from the CPU 60. As the data segments are received, the micro-processor 58 is advised by the micro-processors 50 over the XBUS 64. The micro-processor 58 then asynchronously transfers the data segments into their appropriate location within the assigned buffer space 90. When the entire data of the request is in the assigned buffer space 90, the micro-processor 58 transfers it to the host computer 10 through the FIFO 82 described above. A write operation, of course, operated in much the same manner, except that data flow is in the opposite direction.

Column 11, line 53 – Column 12, line 15

While only shown in simplified representation in FIG. 6, it may be appreciated therefrom and from a consideration of the elements and their method of operation as described above that the single cache memory 56 of substantial size as employed in this embodiment [effect] effects a vast improvement in simplicity and efficiency of operation (i.e. speed). Because of its size, the cache memory 56 will actually self-optimize the data it retains in cache over time so as to minimize the number of data transfers required with the disk drives. In this regard, it is important to recognize that the parity bits associated with data are contained in separately transferable locations on the disk drives 16. Thus, over the course of time the most used data and their parity bits will be virtually permanent residents of the cache memory 56 and will only be written to the disk drives 16 for back-up purposes on a time-available basis in a background mode by the micro-processor 58. The impact of this on the overall performance of the system 22 should not be overlooked or minimized. Consider, if there are eight disk drives 16 containing the individual data bits of data and a ninth disk drive 16 containing the parity bits associated therewith, if the parity bits are contained in an unused portion of the cache memory 56 and only written to the ninth disk drive 16 when the eight disk drives 16 are not otherwise involved in an actual data transfer, disk access time to access data is reduced by one-ninth or eleven percent. The savings in time could be even more substantial in an instance where one of the eight disk drives 16 containing the data bits is malfunctioning and the parity bits are required to correct any errors in the data stream.

B22

Column 14, lines 38-57

In a presently preferred embodiment of the present invention, data is selected for protection by intra-mirroring on the basis of assigning selected Logical Units (LUNs) 100 to be mirrored. That is, one or more Logical Units (LUNs) 100 are selected for mirroring by means of a command identified in FIG. 8 as Logical Unit Mirror Select (LU Mirror Sel) [102C] 107C and the Data Blocks (DBs) 96 containing data that is to be protected by mirroring are assigned Disk Drive 16D storage locations within the selected Logical Units (LUNs) 100. In the present example, by way of illustration, Logical Unit (LUN) 100a is selected by the user for protection by intra-mirroring, and all data that is to be

B23

protected by mirroring is assigned to, that is, stored in, Data Blocks (DBs) 96 having storage addresses, or locations, within Logical Unit (LUN) 100a, that is, in Data Blocks (DBs) 96A through 96K. It will be appreciated by those of ordinary skill in the relevant arts, however, that other methods may be used to select the data to be protected by intra-mirroring, such as by selected address ranges in one or more Logical Units (LUNs) 100.

Column 15, lines 51-67

As illustrated in FIG. 8, the Operating System (OS) 12 programs of Host Computer 10 typically include a System Disk Configuration Utility (SDCU) [100] 99 which controls and manages the initial configuration and formatting of all Disks 16 connected from Host Computer 10. As is well known in the art, the disk drives connected from a host computer can be and are configured and formatted for selected purposes by a system facility such as SDCU [100] 99 at system initialization, and may thereafter be reconfigured and reformatted as necessary, again using SDCU [100.] 99. In the present exemplary system, for example, and as described above, Disk Drives 16 are connected to Host Computer 10 through Disk Platform 28 and Disk Controller 24 and SDCU [100] 99 would, in this implementation, perform the initial configuration and formatting of each of Disk Drives 16 through Disk Controller 24 and Disk Platform 28 in the manner usual in computer systems.

Column 16, lines 1-17

As also indicated in FIG. 8, the allocation of Disk Drives 16 as Disk Drives 16D and 16P is under control of a system user operating through User Input [102,] 107, for example, through a keyboard or user console, and the allocations entered by a user are stored in a Disk Allocation Table [104] 125 which stores, for each Disk Drive 16, its allocation as a Disk Drive 16D or a Disk Drive 16P and the user selection of Logical Units (LUNs) 100, or other subdivisions of the disk storage space, depending upon the implementation of intra-mirroring, to be mirrored. Disk Allocation Table [104] 125 is read at first system initialization by SCDU [100] 99 which, operating through Host Computer 10, Disk Controller 24 and Disk Platform 28, formats and configures each of Disks 16 according to its allocation. The disk allocation process may thereafter be

B25 cont'd

repeated at a later time, for example, as system usage or data storage needs change, whereupon the system will be re-initialized again.

Column 16, lines 25-43

As also illustrated in FIG. 8, Host Computer 10 further includes a Demand Paging and Memory Management Facility (DPMM) 106, wherein, as is usual in such systems and as is well understood in the arts, Demand Paging and Memory Management Facility (DPMM) 106 operates to relate and translate virtual or logical addresses generated by the application and operating systems programs into physical addresses of the corresponding data or program code in the system. As is well understood, the physical addresses may be of locations in system memory or on Disk Drives 16 and, if on Disk Drives 16, will include an identification of the Logical Unit (LUN) 100 and a segment or segments therein for storing one or more Data Blocks (DBs) 96. As is well known in the art, the details of when and where in the system addresses are translated from the logical level to the detailed physical level, and the levels of translation performed at each step, depends upon the distribution of "intelligence" and functions among Disk Controller [25,] 24, Disk Platform 28 and Disk Drives 16 in a particular system.

B26

Column 16, line 63 – Column 17, line 8

As represented in FIG. 8, each Data Address Translation Information (DATI) 114 includes an a Logical Unit Identification (LUNID) 118 identifying the Logical Unit (LUN) 100 that the data item resides in and, from Disk Allocation Table [104,] 125, a Mirror Designator (MD) 120 indicating whether the Logical Unit (LUN) 100 is designated to be intra-mirrored, a Disk Drive Identification (DDI) 122 identifying the particular Disk Drive 16D that at least the first Data Block (DB) 96 of the data item resides in, and, for example, Length Information (LI) 124 to identify the number of Data Blocks (DBs) 96 in the data item so that addresses for all Data Blocks (DBs) 96 of the data item can be generated.

B27

Clumn 17, lines 36-55

In the event of a write operation to a non-mirrored Logical Unit (LUN) 100, each Data Block (DB) 96 comprising the data item will be accompanied by a single Data Block Address (DBA) 126 identifying the locations in Disk Drives 16 that the Data Block (DB) 96 and an address the related parity information is to be written into, together with a write command and the parity information if parity is being written. In the event of a write operation to a mirrored Logical Unit (LUN) 100, however, Demand Paging and Memory Management Facility (DPMM) 106 will note the identification of the Logical [Unit)LUN)] Unit (LUN) as a mirrored Logical Unit (LUN) 100 and will generate and provide two Data Block Addresses (DBAs) 126 with each Data Block (DB) 96, one identifying the location in Disk Drives 16 of the first, or original copy of the Data Block (DB) 96 and the second identifying the location in Disk Drives 16 of the mirror copy of the Data Block (DB) 96. Demand Paging and Memory Management Facility (DPMM) 106 will also provide a write command and parity information with a parity address, if parity information is being written.

B28

Column 18, lines 1-19

In the event of a read operation, Demand Paging and Memory Management Facility (DPMM) 106 will operate as described above with respect to generating Data Block Addresses (DBAs) 126 to non-mirrored Logical Units (LUNs) 100, but will receive rather than provide a Data Block (DB) 96 with the related parity information for each Data Block Address (DBA) 126. In the event of a read from a mirrored Logical Unit (LUN) [96] 100, Demand Paging and Memory Management Facility (DPMM) 106 will generate a read address in the manner described above with respect to a write address and will issue the address and a read command to Disk Controller 24, together with a corresponding parity address. If the parity check reveals an unrecoverable error in the data read from the first copy of the Data Block (DB) 96, Demand Paging and Memory Management Facility (DPMM) 106 will generate a second read address to the duplicate copy of the Data Block (DB) 96, with a corresponding parity address, and will thereby read the mirrored copy of the Data Block (DB) 96.

B29

Column 18, lines 49-56

[While this] This method of caching data and parity information to be written to and read from the disk drives may reduce the typical data write time by approximately 10%. There are circumstances, however, when it is desirable to reduce the new data write time even further, as when the system receives large amounts of new data at short intervals or when the time to acquire new data is severely limited, as may occur when data is downloaded from a satellite link.

B30

Column 18, line 57 – Column 19, line 7

According to the present invention, therefore, and by selection through a user command entered through User Input [102,] 107, identified in FIG. 8 as Parity Generate De-Assert (ParityGenDA) [102A,] 107A, Demand Paging and Memory Management Facility (DPMM) 106 may be directed to neither generate nor write parity information when writing data to Disk Drives 16. As such, and while the mass storage system of the present invention will write Data Blocks (DBs) 96 to Disk Drives 16D in the manner described above, the generation and writing of parity information will be inhibited, so that no Parity Blocks (PBs) 98 are written to the Disk Drives 16 allocated for parity information. As a consequence, the representations of data and parity information stored on Disk Drives 16 and illustrated in FIGS. 7A and 7B will appear as previously discussed herein above, except that Disk Drive(s) 16P will contain no Parity Blocks (PBs) 98 relating to the Data Blocks (DBs) 96 written while operating in the parity inhibited mode.

B31

Column 19, lines 8-19

Thereafter, at the control of the user and generally when the rate at which new data must be written to Disk Drives 16 has returned to lower levels, the user may enter a command through User Input [102,] 107, identified in FIG. 8 as Parity Generate Assert (ParityGenA) [102B,] 107B, directing the system to generate and store parity information for the Data Blocks (DBs) 96 which have been written to Disk Drives 16 without corresponding Parity Blocks (PBs) 98. In response, Demand Paging and Memory Management Facility (DPMM) 106 will first determine which Data Blocks (DBs) 96 have been written without corresponding Parity Blocks (PBs) 98, which may be

B32

B32 cont'd

accomplished in a number of ways, as will appreciated by those of ordinary skill in the relevant arts.

Column 19, lines 60-67

B33

It will be appreciated by those [or] <u>of</u> ordinary skill in the arts that while Parity Generator 128 is illustrated in FIG. 8 as associated with Demand Paging and Memory Management Facility (DPMM) 106 in Host Computer 10, Parity Generator [126] <u>128</u> may be implemented in a number of ways in the mass storage system of the present invention. For example, Parity Generator [126] <u>128</u> may also be implemented in Disk Controller 24 or in Disk Platform 28.

Column 20, lines 9-31

B34

It will also be understood that the data protection mechanisms of the present invention as described above allow a user to selectively tailor the level of protection in a mass storage system to meet the changing requirements of operation and use of the system by providing selectable levels of protection. For example, at the lowest level the system may be operated without mirroring and with parity generation and storing inhibited, thereby providing the maximum rate of data reception and storage but the lowest level of protection. At the next level, parity information may be generated and stored while the data is being received and stored, thus providing a higher level of protection but a lower data transfer rate. At the next level, the user may select intra-mirroring, wherein data that is selected and identified by the user to be additionally protected is written to logical units of the disk drives that have been selected and designated as mirrored logical units, thereby providing a still higher level of protection for the most critical or important data at the cost of some increase in storage space. Finally, [the] using the same mechanism as provided for intra-mirroring but designating all logical units for mirroring, the user may achieve full mirroring for all data stored in the system, but at the cost of significantly increased storage space.